

8. Der Äquivalenzsatz

ÄQUIVALENZSATZ. Für eine (partielle) Funktion $f : \mathbb{N}^n \rightarrow \mathbb{N}$ sind folgende Aussagen äquivalent:

- f ist (partiell) Turing-berechenbar.
- f wird von einer k -Band-Turingmaschine berechnet ($k \geq 1$).
- f wird von einem Turingoperator berechnet.
- f wird von einer Registermaschine berechnet.
- f wird von einem Registeroperator (konservativ) berechnet.
- f ist (partiell) rekursiv.

D.h. es gilt

$$\begin{aligned} F(\text{TM}) &= F(k\text{-TM}) = \bigcup_{k \geq 1} F(k\text{-TM}) = F(\text{TO}) \\ &= F(\text{RM}) = F(\text{RO}) = F_{kon}(\text{RO}) = F(\text{REK}). \end{aligned}$$

In den vorangegangenen Abschnitten haben wir bereits die Inklusionen

$$\begin{aligned} F(\text{REK}) \subseteq F_{kon}(\text{RO}) = F(\text{RO}) \subseteq F(\text{TO}) \\ \subseteq F(\text{TM}) = F(k\text{-TM}) = \bigcup_{k \geq 1} F(k\text{-TM}) \end{aligned}$$

gezeigt sowie (z.T. in den Übungen)

$$F(\text{RO}) \subseteq F(\text{RM}) \subseteq \bigcup_{k \geq 1} F(k\text{-TM})$$

Zum Beweis des Äquivalenzsatzes genügt es also noch zu zeigen:

$$F(\text{TM}) \subseteq F(\text{REK})$$

Der BEWEIS wird auf den folgenden Folien skizziert.

Sei $\varphi^{(n)} \in F(\text{TM})$. Um $\varphi^{(n)} \in F(\text{REK})$ zu zeigen gehen wir wie folgt vor:

- (1) NORMIERUNG: Wähle eine geeignet normierte TM M , die φ berechnet.
- (2) GÖDELISIERUNG (KODIERUNG): Kodiere die Maschine M , M -Konfigurationen und M -Rechnungen durch Zahlen so, dass das Prädikat

$$\text{rechnung}(\vec{x}, y, z) \Leftrightarrow y \text{ kodiert eine terminierende } M\text{-Rechnung} \\ \text{bei Eingabe } \vec{x} \text{ mit Ergebnis } z$$

primitiv rekursiv ist.

Dann gilt

$$\varphi(\vec{x}) = (\mu y(\text{rechnung}(\vec{x}, (y)_1, (y)_2)))_2,$$

womit $\varphi \in F(\text{REK})$ bewiesen ist.

NORMIERUNG

Es sei $M = (\{0\}, n, \{0\}, \Gamma, Z, z_0, \delta)$ eine Turingmaschine, die φ berechnet.

O.B.d.A. können wir folgende Annahmen machen:

- $\Gamma = \{0, 1 = b, \dots, p\}$ (für $p \geq 1$ geeignet)
- $Z = \{0, 1, \dots, q\}$ (für $q \geq 1$ geeignet)
- $z_0 = 0$
- $z_{stopp} := q$ ist ausgezeichneter Stoppzustand von M

Buchstaben des Bandalphabets und Zustände sind also Zahlen.

Wir erreichen dies durch Umbenennen der Buchstaben des Bandalphabets (die nicht zum Ein- und Ausgabealphabet gehören) und der Zustände. Den Stoppzustand q erhalten wir dadurch, dass wir diesen Zustand zur ursprünglichen Zustandsmenge hinzunehmen und für jedes Paar (z, a) , für das δ nicht definiert ist, die Programmzeile $\delta(z, a) = (a, S, q)$ neu hinzufügen.

GÖDELISIERUNG

Die folgende Kodierung der Bestandteile und Arbeitsweise von M benutzt die im letzten Kapitel eingeführte primitiv rekursive Kodierung endlicher Zahlenfolgen. Wir erinnern an folgende Notationen und Funktionen:

$$\begin{aligned}\langle x_1, \dots, x_n \rangle &:= \tau^*(x_1, \dots, x_n) \\ \langle \rangle &:= \tau^*(\lambda) \\ (x)_i &:= \pi^*(x, i) \quad \text{d.h.} \quad (\langle x_1, \dots, x_n \rangle)_i = x_i \\ (x)_{i,j} &:= \pi^*(\pi^*(x, i), j) \\ l(\langle x_1, \dots, x_n \rangle) &= n \\ \text{head}(\langle x_1, \dots, x_n \rangle) &= x_1 \\ \text{tail}(\langle x_1, \dots, x_n \rangle) &= \langle x_2, \dots, x_n \rangle \\ \langle x_1, \dots, x_n \rangle \circ \langle y_1, \dots, y_m \rangle &= \langle x_1, \dots, x_n, y_1, \dots, y_m \rangle\end{aligned}$$

KODIERUNG VON KONFIGURATIONEN

Einer Konfiguration

$$c = \dots a_{-l} \dots a_{-1} a_0 \dots a_r \dots$$

\uparrow
 z

ordnen wir die Gödelnummer

$$\tilde{c} = \langle z(c), l(c), r(c) \rangle$$

zu, wobei

$$\begin{aligned} z(c) &= z && \text{(Zustand)} \\ l(c) &= \langle a_{-1}, \dots, a_{-l} \rangle && \text{(linke Bandhälfte; von rechts gelesen)} \\ r(c) &= \langle a_0, \dots, a_r \rangle && \text{(rechte Bandhälfte inkl. AF; von links gelesen).} \end{aligned}$$

NB(1): Da die Buchstaben des Bandalphabets Zahlen sind, können wir hier ein Wort $a_1 \dots a_n$ durch die kodierte Zahlenfolge $\langle a_1, \dots, a_n \rangle$ beschreiben.

NB(2): Die wesentliche Information über c lässt sich primitiv rekursiv aus \tilde{c} extrahieren. Z.B. ist $(\tilde{c})_1$ der Zustand von c , $(\tilde{c})_2$ und $(\tilde{c})_3$ die kodierte linke bzw. rechte Bandhälfte und $(\tilde{c})_{3,1}$ die Inschrift des AF. Zur leichteren Lesbarkeit schreiben wir:

$$z(x) := (x)_1 \quad \text{lbh}(x) := (x)_2 \quad \text{rbh}(x) := (x)_3 \quad \text{af}(x) := (x)_{3,1}$$

Als nächstes zeigen wir, dass folgende Funktionen primitiv rekursiv sind:

(1) start: ordnet einer Eingabe die Gödelnummer der zugehörigen Startkonfiguration zu

(2) wert: ordnet der Gödelnummer einer Konfiguration die zugehörige Ausgabe zu

(3) nfk: ordnet der Gödelnummer einer Nichtstopp-Konfiguration die Gödelnummer der Nachfolgekongfiguration zu

Hiermit werden wir dann die primitive Rekursivität des Rechnungsprädikates rechnerisch nachweisen.

KODIERTE EINGABEFUNKTION

BEHAUPTUNG. Die Funktion $\text{start} : \mathbb{N}^n \rightarrow \mathbb{N}$ mit

$$\text{start}(\vec{x}) = \text{start}(x_1, \dots, x_n) = \tilde{c},$$

wobei c die zu \vec{x} gehörende Startkonfiguration

$$c = \dots \underset{\uparrow}{1} \underline{1} \underline{x_1} \underline{1} \underline{x_2} \dots \underline{1} \underline{x_n} \underline{1} \dots$$

ist, ist primitiv rekursiv.

BEWEIS. Definiere start durch

$$\text{start}(\vec{x}) = \langle 0, \text{lbhstart}(\vec{x}), \text{rbhstart}(\vec{x}) \rangle$$

wobei

$$\begin{aligned} \text{lbhstart}(\vec{x}) &= \langle 1 \rangle \\ \text{rbhstart}(\vec{x}) &= \langle 1 \rangle \circ \text{unär}(x_1) \circ \langle 1 \rangle \circ \dots \circ \text{unär}(x_n) \circ \langle 1 \rangle \end{aligned}$$

Hierbei ordnet die durch

$$\begin{aligned} \text{unär}(0) &= \langle 0 \rangle \\ \text{unär}(m+1) &= \text{unär}(m) \circ \langle 0 \rangle \end{aligned}$$

definierte Funktion unär einer Zahl m die kodierte Folge $\langle \underline{m} \rangle = \langle 0, \dots, 0 \rangle$ der Länge $m+1$ zur Beschreibung der Unärdarstellung 0^{m+1} von m zu.

KODIERTE AUSGABEFUNKTION

BEHAUPTUNG. Die Funktion $\text{wert} : \mathbb{N} \rightarrow \mathbb{N}$, die der Kodierung \tilde{c} einer Konfiguration c den zugehörige Ausgabewert zuordnet, d.h. $\text{wert}(\tilde{c}) = m$ für

$$c = \dots a_{-l} \dots a_{-1} a_0 \underline{m} a_{m+2} \dots a_r \dots \quad (a_{m+2} \neq 0),$$

\uparrow
 z

ist primitiv rekursiv.

BEWEIS. Da $m + 2$ durch die Länge der relevanten rechten Bandhälfte beschränkt ist und a_{m+2} der erste rechts des Arbeitsfeldes stehende von 0 verschiedene Buchstabe ist, gilt

$$\text{wert}(x) = \mu y \leq l(\text{rbh}(x)) (y \geq 2 \ \& \ (\text{rbh}(x))_y \neq 0) \dot{-} 2.$$

KODIERTE EINSCHRITTFUNKTION

BEHAUPTUNG. Die Funktion $\text{nfk} : \mathbb{N} \rightarrow \mathbb{N}$, die der Gödelnummer \tilde{c} einer Nichtstoppkonfiguration c die Gödelnummer der Nachfolgekonfiguration von c zuordnet (und die die Gödelnummer \tilde{c} einer Stoppkonfiguration c auf sich selbst abbildet), ist primitiv rekursiv.

BEWEIS. Wir setzen

$$\text{nfk}(x) = \langle \text{nfz}(x), \text{nflbh}(x), \text{nfrbh}(x) \rangle$$

aus den im Folgenden definierten primitiv rekursiven Hilfsfunktionen $\text{nfz}(x)$, $\text{nflbh}(x)$ und $\text{nfrbh}(x)$ zusammen, die bei Eingabe $x = \tilde{c}$ Zustand sowie kodierte linke bzw. rechte Bandhälfte der Nachfolgekonfiguration c' von c ausgeben.

Zur Definition dieser 3 Komponenten von nfk stellen wir das Programm δ zunächst durch drei primitiv rekursive Funktionen dar:

- $nz(z, a)$ gibt den Nachfolgezustand des Zustandes z an, falls a auf dem Arbeitsfeld steht:

$$nz(z, a) = \begin{cases} z' & \text{falls } \delta(z, a) = (a', B, z') \text{ (für } a', B \text{ geeignet)} \\ z & \text{sonst} \end{cases}$$

- $no(z, a)$ gibt den im Zustand z ausgeführten Druckbefehl an, falls a auf dem Arbeitsfeld steht:

$$no(z, a) = \begin{cases} a' & \text{falls } \delta(z, a) = (a', B, z') \text{ (für } B, z' \text{ geeignet)} \\ a & \text{sonst} \end{cases}$$

- $nb(z, a)$ gibt den im Zustand z ausgeführten Bewegungsbefehl an, falls a auf dem Arbeitsfeld steht, wobei $\tilde{L} = 0$, $\tilde{S} = 1$ und $\tilde{R} = 2$:

$$nb(z, a) = \begin{cases} \tilde{B} & \text{falls } \delta(z, a) = (a', B, z') \text{ (für } a', z' \text{ geeignet)} \\ \tilde{S} & \text{sonst} \end{cases}$$

Diese Funktionen sind primitiv rekursiv, da $nz(z, a) \stackrel{*}{=} z$, $no(z, a) \stackrel{*}{=} a$ und $nb(z, a) \stackrel{*}{=} 1$.

Hiermit können wir nun definieren:

$$\text{nfz}(x) := \text{nz}(z(x), \text{af}(x))$$

$$\text{nflbh}(x) := \begin{cases} \text{lbh}(x) & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{S} \\ \text{tail}(\text{lbh}(x)) \circ \langle 1 \rangle & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{L} \\ \langle \text{no}(z(x), \text{af}(x)) \rangle \circ \text{lbh}(x) & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{R} \end{cases}$$

$$\text{nfrbh}(x) := \begin{cases} \langle \text{no}(z(x), \text{af}(x)) \rangle \circ \text{tail}(\text{rbh}(x)) & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{S} \\ \text{head}(\text{lbh}(x)) \circ \langle \text{no}(z(x), \text{af}(x)) \rangle \circ \text{tail}(\text{rbh}(x)) & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{L} \\ \text{tail}(\text{rbh}(x)) \circ \langle 1 \rangle & \text{falls } \text{nb}(z(x), \text{af}(x)) = \tilde{R} \end{cases}$$

Bei der Definition von nflbh erweitern wir hierbei den relevanten Teil der linken Bandhälfte links um ein Blank, falls diese durch den L -Befehl rechts um ein Feld gekürzt wird (s. Fall 2). Analog bei nfrbh , weshalb die relevanten Bandteile nie leer sind.

Weiter beachte man, dass für ein x , das keine Konfiguration mit Zustand $z < q$ kodiert, $\text{nfk}(x) = \langle \text{nfz}(x), \text{nflbh}(x), \text{nfrbh}(x) \rangle = x$ gilt.

KODIERUNG VON KONFIGURATIONENFOLGEN

Eine Folge c_1, \dots, c_m von Konfigurationen stellen wir als kodierte Folge der kodierten Konfigurationen dar: $\langle \tilde{c}_1, \dots, \tilde{c}_m \rangle$

BEHAUPTUNG. Es gibt ein 1-stelliges primitiv rekursives Prädikat tkf, das auf eine Zahl x , deren erste (kodierte) Komponente $(x)_1$ eine Nichtstoppkonfiguration kodiert, genau dann zutrifft, falls x eine maximale endliche M -Konfigurationenfolge kodiert:

$$(x)_1 = \tilde{c} \quad \& \quad c \text{ Nichtstoppkonfiguration} \quad \Rightarrow$$

$$[\text{tkf}(x) \Leftrightarrow x \text{ kodiert eine endl. maximale } M\text{-Konfigurationenfolge} \\ c_1, \dots, c_m, \quad \text{d.h.} \\ x = \langle \tilde{c}_1, \dots, \tilde{c}_m \rangle \quad \& \quad c_1 \Rightarrow_M c_2 \cdots \Rightarrow_M c_m \quad \& \quad c_m \text{ Stoppkonfiguration}]$$

BEWEIS. Definiere tkf durch

$$\text{tkf}(x) \quad :\Leftrightarrow \quad l(x) \geq 1 \quad \& \\ \forall i < l(x) \quad (i \geq 1 \Rightarrow z((x)_i) < q \quad \& \quad \text{nfk}((x)_i) = (x)_{i+1}) \quad \& \\ z((x)_{l(x)}) = q$$

KODIERTES RECHNUNGSPRÄDIKAT

BEHAUPTUNG. Das $(n+2)$ -stellige Prädikat rechnung mit

$$\text{rechnung}(\vec{x}, y, z) \Leftrightarrow y \text{ kodiert eine terminierende } M\text{-Rechnung} \\ \text{bei Eingabe } \vec{x} \text{ mit Ergebnis } z$$

ist primitiv rekursiv.

BEWEIS.

$$\text{rechnung}(\vec{x}, y, z) \Leftrightarrow \text{start}(\vec{x}) = (y)_1 \ \& \ \text{tkf}(y) \ \& \ \text{wert}((y)_{l(y)}) = z$$

Wie bereits bemerkt, folgt hieraus wegen

$$\varphi(\vec{x}) = (\mu y(\text{rechnung}(\vec{x}, (y)_1, (y)_2)))_2,$$

dass φ partiell rekursiv ist. *q.e.d.*

ÄQUIVALENZSATZ UND CHURCH-TURING-THESE

Neben den drei von uns betrachteten Ansätzen zur Formalisierung des Berechenbarkeitsbegriffs wurde eine Reihe anderer formaler Berechnungskonzepte (z.B. *Markov-Algorithmen*, der *λ -Kalkül* und *Post's Kanonische Systeme*) eingeführt, die sich ebenfalls als äquivalent zur Turing-Berechenbarkeit erwiesen haben.

Man geht daher davon aus, dass mit der Turing-Berechenbarkeit eine adäquate Formalisierung des intuitiven Berechenbarkeitsbegriffs vorliegt. Dies ist Inhalt der CHURCH-TURING-THESE (manchmal auch kurz CHURCHSCHE THESE genannt).

CHURCH-TURING-THESE FÜR DIE ENTSCHEIDBARKEIT

Nach der Church-Turing-These und dem Äquivalenzsatz stimmen die (partiell) berechenbaren Funktionen mit den (partiell) rekursiven Funktionen überein.

Da die entscheidbaren Mengen gerade die Mengen sind, deren charakteristische Funktion berechenbar ist, und wir die rekursiven Mengen entsprechend als die Mengen definiert haben, deren charakteristische Funktion rekursiv ist, impliziert die Church-Turing-These auch die Äquivalenz

$$A \text{ entscheidbar} \Leftrightarrow A \text{ rekursiv}$$

CHURCH-TURING-THESE FÜR DIE AUFZÄHLBARKEIT

Den Begriff der Aufzählbarkeit konnten wir auf den der Berechenbarkeit zurückführen, indem wir beobachteten, dass eine Menge genau dann aufzählbar ist, wenn sie der Definitionsbereich einer partiell berechenbaren Funktion ist. Wir definieren nun entsprechend:

DEFINITION. Eine Menge $M \subseteq \mathbb{N}^n$ ist *rekursiv aufzählbar (r.a.)*, wenn M der Definitionsbereich einer partiell rekursiven Funktion $\varphi : \mathbb{N}^n \rightarrow \mathbb{N}$ ist.

Aus der Church-Turing-These für die Berechenbarkeit ergibt sich dann:

$$A \text{ aufzählbar} \Leftrightarrow A \text{ rekursiv aufzählbar}$$

EINE FOLGERUNG AUS DEM ÄQUIVALENZSATZ

Die im Beweis des Äquivalenzsatzes gefundene Darstellung von φ zeigt, dass jede partiell rekursive Funktion eine Darstellung hat, in der der μ -Operator nur einmal angewendet wird, während die anderen Bestandteile der Darstellung primitiv rekursiv sind:

$$\varphi^{(n)}(\vec{x}) = f^{(1)}(\mu y(P^{(n+1)}(\vec{x}, y)))$$

wobei $f, P \in F(\text{PRIM})$.

Hiermit kann man zeigen:

SATZ. Für eine totale Funktion $f^{(n)}$ über \mathbb{N} sind folgende Aussagen äquivalent:

- (a) f ist primitiv rekursiv.
- (b) f wird von einer Turingmaschine M berechnet, deren Laufzeit $\text{time}_M(\vec{x})$ durch eine primitiv rekursive Funktion $t(\vec{x})$ beschränkt ist.

BEWEISIDEE. (a) \Rightarrow (b) Induktion nach dem Aufbau der primitiv rekursiven Funktionen. (b) \Rightarrow (a) Nach den vorangehenden Bemerkungen genügt es zu zeigen, dass die kodierte Rechnung einer Turingmaschine primitiv rekursiv in der Eingabe und in der Rechenzeit beschränkt ist. Dies ergibt sich leicht aus der Definition der kodierten Rechnung. (Übung!)

KLEINER ÄQUIVALENZSATZ

Weiter lassen sich die primitiv rekursiven Funktionen mit Hilfe der primitiven Registeroperatoren beschreiben:

SATZ. $F(\text{PRO}) = F(\text{PRIM})$.

Da wir bereits $F(\text{PRIM}) \subseteq F(\text{PRO}) = F_{kon}(\text{PRO})$ gezeigt haben, genügt es hierzu folgendes Lemma zu beweisen.

LEMMA. $F_{kon}(\text{PRO}) \subseteq F(\text{PRIM})$.

Aus Zeitgründen verzichten wir auf den Beweis, der ebenfalls mit Kodierungen arbeitet: Hier werden Registermaschinen gödelisiert.

Bei Interesse kann der Beweis im Skript (Kapitel 10) nachgelesen werden:
<http://www.math.uni-heidelberg.de/logic/skripten.html>

STICHWORT GÖDELISIERUNG

Kurt Gödel 1906 (Brünn/Brno) - 1978 (Princeton)

Bedeutendster Mathematischer Logiker des 20. Jahrhunderts

Gödel benutzte die Technik der Gödelisierung um seine berühmten Unvollständigkeitssätze zu zeigen.

1. UNVOLLSTÄNDIGKEITSSATZ. Es gibt keinen Kalkül der Arithmetik, in dem alle wahren Sätze der Arithmetik herleitbar sind.

2. UNVOLLSTÄNDIGKEITSSATZ. Die Konsistenz eines Kalküls lässt sich nicht mit Mitteln des Kalküls alleine beweisen.

(\Rightarrow Hilbert's Programm zur Formalisierung der Mathematik ist undurchführbar)

Berechenbarkeitstheoretische Version des 1. Unvollständigkeitssatzes:

- Die in einem Kalkül herleitbaren Sätze bilden eine rekursiv aufzählbare Menge.
- Die Menge der wahren Sätzen der Arithmetik ist nicht rekursiv aufzählbar.

(Hierbei wird von einer Gödelisierung der Formeln ausgegangen.)

Unser erstes Ziel, nämlich den Algorithmenbegriff zu formalisieren, haben wir erreicht. Wir benutzen dies nun um

- Grundergebnisse der Berechenbarkeitstheorie
(z.B. Existenz universeller Maschinen)
- Grenzen der Berechenbarkeit
(z.B. Unentscheidbarkeit des Halteproblems)

aufzuzeigen.